**Amazon API Gateway FAQs**

## **General**

Q: What is Amazon API Gateway?

Amazon API Gateway is a fully managed service that makes it easy for developers to publish, maintain, monitor, and secure APIs at any scale. With a few clicks in the AWS Management Console, you can create an API that acts as a “front door” for applications to access data, business logic, or functionality from your back-end services, such as applications running on Amazon Elastic Compute Cloud (Amazon EC2), code running on AWS Lambda, or any web application. Amazon API Gateway handles all of the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, authorization and access control, monitoring, and API version management. Amazon API Gateway has no minimum fees or startup costs. You pay only for the API calls you receive and the amount of data transferred out.

Q: Why use Amazon API Gateway?

Amazon API Gateway provides developers with a simple, flexible, fully managed, pay-as-you-go service that handles all aspects of creating and operating robust APIs for application back ends.&nbsp; With Amazon API Gateway, you can launch new services faster and with reduced investment so you can focus on building your core business services.&nbsp; Amazon API Gateway was built to help you with several aspects of creating and managing APIs:

1) Metering. API Gateway helps you define plans that meter and restrict third-party developer access to your APIs. You can define a set of plans, configure throttling, and quota limits on a per API key basis. API Gateway automatically meters traffic to your APIs and lets you extract utilization data for each API key.

2) Security. API Gateway provides you with multiple tools to authorize access to your APIs and control service operation access. Amazon API Gateway allows you to leverage AWS administration and security tools, such as AWS Identity and Access Management (IAM) and Amazon Cognito, to authorize access to your APIs. Amazon API Gateway can verify signed API calls on your behalf using the same methodology AWS uses for its own APIs. Using Lambda authorizers written as AWS Lambda functions, API Gateway can also help you verify incoming bearer tokens, removing authorization concerns from your backend code.

3) Resiliency. Amazon API Gateway helps you manage traffic with throttling so that backend operations can withstand traffic spikes. Amazon API Gateway also helps you improve the performance of your APIs and the latency your end users experience by caching the output of API calls to avoid calling your backend every time.

4) Operations Monitoring. After an API is published and in use, API Gateway provides you with a metrics dashboard to monitor calls to your services. The Amazon API Gateway dashboard, through integration with Amazon CloudWatch, provides you with backend performance metrics covering API calls, latency data and error rates. You can enable detailed metrics for each method in your APIs and also receive error, access or debug logs in CloudWatch Logs.

5) Lifecycle Management. After an API has been published, you often need to build and test new versions that enhance or add new functionality. Amazon API Gateway lets you operate multiple API versions and multiple stages for each version simultaneously so that existing applications can continue to call previous versions after new API versions are published.

6) Designed for Developers. Amazon API Gateway allows you to quickly create APIs and assign static content for their responses to reduce cross-team development effort and time-to-market for your applications. Teams who depend on your APIs can begin development while you build your backend processes.

Q: How do I get started with Amazon API Gateway?

You can quickly and easily create a custom API using Amazon API Gateway. For a simple “Hello World” example, follow these steps:

1. Go to the Amazon API Gateway console.

2. Select an existing REST API or create a new one by entering a name for the API.

3. On the REST API tree view, click “Create Resource”.

4. Choose a name for your resource, such as “cars”.

5. With the new resource selected, click the button to create a new method and select the HTTP verb associated with the method (for example, GET).

6. Select the integration type (for example, HTTP Proxy), and enter the URL the Amazon API Gateway should call.

7. Define how requests and responses are transformed using a mapping template, or accept the default settings to pass all of the request and response data through without applying any transformation.

8. Configure the method’s security settings.

9. Deploy your new API to a stage.

10. From the Stage management page, set up caching and throttling.

11. On the Client Platforms tab in the Amazon API Gateway console, click the button to download the Android, iOS SDK, or JavaScript library that contains helper methods to call your sayHello operation. The SDK library makes calling your APIs similar to calling a local method. The client SDK automatically handles retries, informing the developer of network or other fault conditions. The SDK library includes the logic necessary to authenticate the client application to your APIs.

12. Integrate the downloaded SDK into your mobile application. Write the code to invoke your custom API. For example, to invoke the getCar(int carId) API in an iOS application:

–(void)getSampleCar

{

NSString \*response = [MyServiceClient getCar:1323];

NSLog( @”Response was [%@]”, response );

}

13.Run your application.

Q: Can I create HTTPS endpoints?

Yes, all of the APIs created with Amazon API Gateway expose HTTPS endpoints only. Amazon API Gateway does not support unencrypted (HTTP) endpoints. By default, Amazon API Gateway assigns an internal domain to the API that automatically uses the Amazon API Gateway certificate. When configuring your APIs to run under a custom domain name, you can provide your own certificate for the domain.

Q: What data types can I use with Amazon API Gateway ?

APIs built on Amazon API Gateway can accept any payloads sent over HTTP. Typical data formats include JSON, XML, query string parameters, and request headers. You can declare any content type for your API’s responses, and then use the transform templates to change the back-end response into your desired format.

Q: With what backends can Amazon API Gateway communicate?

Amazon API Gateway can execute AWS Lambda functions in your account, start AWS Step Functions state machines, or call HTTP endpoints hosted on AWS Elastic Beanstalk, Amazon EC2, and also non-AWS hosted HTTP based operations that are accessible via the public Internet.&nbsp;API Gateway also allows you to specify a mapping template to generate static content to be returned, helping you mock your APIs before the backend is ready. You can also integrate API Gateway with other AWS services directly – for example, you could expose an API method in API Gateway that sends data directly to Amazon Kinesis.

Q: For which client platforms can Amazon API Gateway generate SDKs?

API Gateway generates custom SDKs for mobile app development with Android and iOS, and for web app development with JavaScript. Once an API and its models are defined in API Gateway, you can use the AWS console or the API Gateway APIs to generate and download a client SDK.

Q: In which AWS regions is Amazon API Gateway available?

Please refer to [Regional Products and Services](https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/) for details of Amazon API Gateway service availability by region.

Q: What can I manage through the Amazon API Gateway console?

Through the Amazon API Gateway console, you can define the REST API and its associated resources and methods, manage the API lifecycle, generate client SDKs and view API metrics. You can also use the API Gateway console to define your APIs’ usage plans, manage developers’ API keys, and configure throttling and quota limits. All of the same actions are available through the API Gateway APIs.

Q: What is a REST API?

In Amazon API Gateway, a REST API is a group of resources and methods, or endpoints. REST APIs can be deployed to different stages and cloned to new versions.

Q: What is a resource?

A resource is a typed object that is part of your API’s domain. Each resource may have associated a data model, relationships to other resources, and can respond to different methods.You can also define resources as variables to intercept requests to multiple child resources.

Q: What is a method?

Each resource within a REST API can support one or more of the standard HTTP methods. You define which verbs should be supported for each resource (GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS) and their implementation. For example, a GET to the cars resource should return a list of cars. To connect all methods within a resource to a single backend endpoint, API Gateway also supports a special “ANY” method.

Q: What is an usage plan?  
  
Usage plans help you declare plans for third-party developers that restrict access only to certain APIs, define throttling and request quota limits, and associate them with API keys. You can also extract utilization data on an per-API key basis to analyze API usage and generate billing documents. For example, you can create a basic, professional, and enterprise plans – you can configure the basic usage plan to only allow 1,000 requests per day and a maximum of 5 requests per second (RPS).

Q: What is the Amazon API Gateway API lifecycle?

With Amazon API Gateway, each REST API can have multiple stages. Stages are meant to help with the development lifecycle of an API -- for example, after you’ve built your APIs and you deploy them to a development stage, or when you are ready for production, you can deploy them to a production stage.

Q: What is a stage?

In Amazon API Gateway, stages are similar to tags. They define the path through which the deployment is accessible. For example, you can define a development stage and deploy your cars API to it. The resource will be accessible at https://www.myapi.com/dev/cars. You can also set up custom domain names to point directly to a stage, so that you don’t have to use the additional path parameter. For example, if you pointed myapi.com directly to the development stage, you could access your cars resource at https://www.myapi.com/cars. Stages can be configured using variables that can be accessed from your API configuration or mapping templates.

Q: What are stage variables?

Stage variables let you define key/value pairs of configuration values associated with a stage. These values, similarly to environment variables, can be used in your API configuration. For example, you could define the HTTP endpoint for your method integration as a stage variable, and use the variable in your API configuration instead of hardcoding the endpoint – this allows you to use a different endpoint for each stage (e.g. dev, beta, prod) with the same API configuration. Stage variables are also accessible in the mapping templates and can be used to pass configuration parameters to your Lambda or HTTP backend.

Q: What if I mistakenly deployed to a stage?

Amazon API Gateway saves the history of your deployments. At any point, using the Amazon API Gateway APIs or the console, you can roll back a stage to a previous deployment.

Q: Can I run multiple versions of the same REST API?

Yes. Amazon API Gateway gives you the ability to clone an existing API. When you are ready to start working on the next major version of your API, you will be able to keep working on your version 1 and version 2 APIs simultaneously.

Can Amazon API Gateway work within an Amazon VPC?

No. Amazon API Gateway endpoints are always public to the Internet. However, you can expose your HTTP/HTTPS resources in your Amazon VPC by setting up [Private Integrations](https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-private-integration.html) using VPC Links. You can also generate a client-side SSL certificate in Amazon API Gateway to verify that requests to your backend systems were sent by API Gateway using the public key of the certificate.

Q: Can I use my Swagger API definitions?

Yes. You can use our open source [Swagger importer tool](https://github.com/awslabs/aws-apigateway-importer) to import your Swagger API definitions into Amazon API Gateway. With the Swagger importer tool you can create and deploy new APIs as well as update existing ones.

Q: How do I monetize my APIs on API Gateway?

You can monetize your APIs on API Gateway by publishing them as products in AWS Marketplace. You will first need to [register as a seller](https://signin.aws.amazon.com/signin?redirect_uri=https%3A%2F%2Faws.amazon.com%2Fmarketplace%2Fmanagement%2Fregister%2F%3Fstate%3DhashArgs%2523%26isauthcode%3Dtrue&client_id=arn%3Aaws%3Aiam%3A%3A015428540659%3Auser%2Faws-mp-seller-management-portal&forceMobileApp=0) in AWS Marketplace, and submit your usage plans on API Gateway as products. [Read here](https://aws.amazon.com/blogs/compute/monetize-your-apis-in-aws-marketplace-using-api-gateway/) to learn more about API Monetization.

Q: How do I document my API on Amazon API Gateway?

API Gateway offers the ability to create, update, and delete documentation associated with each portion of your API, such as methods and resources. You can access documentation-related APIs through the AWS SDKs, CLI, via RESTful calls, or by editing the documentation strings directly in the API Gateway console. Documentation can also be imported as a Swagger file, either as part of the API or separately, allowing you to add or update the documentation without disturbing the API definition. API Gateway conforms to the [Open API specification](https://swagger.io/specification/) for documentation imported from, or exported to, Swagger files.

Q: How can I avoid creating redundant copies of error messages and other documentation that recurs frequently in my API?

In addition to offering standards-conformant API documentation support, API Gateway additionally supports documentation inheritance, making it simple to define a documentation string once and then use it in multiple places. Inheritance simplifies the process of defining API documentation, and can be converted to the standard representation when exporting the API as a Swagger file.

[back to top >>](https://aws.amazon.com/api-gateway/faqs/)

## **Security and Authorization**

Q: How do I authorize access to my APIs?

With Amazon API Gateway, you can optionally set your API methods to require authorization. When setting up a method to require authorization you can leverage AWS Signature Version 4 or Lambda authorizers to support your own bearer token auth strategy.

Q: How does AWS Signature Version 4 work?

You can use AWS credentials - access and secret keys - to sign requests to your service and authorize access like other AWS services. The signing of an Amazon API Gateway API request is managed by the custom API Gateway SDK generated for your service. You can retrieve temporary credentials associated with a role in your AWS account using Amazon Cognito.

Q: What is a Lambda authorizer?

Lambda authorizers are AWS Lambda functions. With custom request authorizers, you will be able to authorize access to APIs using a bearer token auth strategy such as OAuth. When an API is called, API Gateway checks if a Lambda authorizer is configured, API Gateway then calls the Lambda function with the incoming authorization token. You can use Lambda to implement various authorization strategies (e.g. JWT verification, OAuth provider callout) that return IAM policies which are used to authorize the request. If the policy returned by the authorizer is valid, API Gateway will cache the policy associated with the incoming token for up to 1 hour.

Q: Can Amazon API Gateway generate API keys for distribution to third-party developers?

Yes. API Gateway can generate API keys and associate them with an usage plan. Calls received from each API key are monitored and included in the Amazon CloudWatch Logs you can enable for each stage. However, we do not recommend you use API keys for authorization. You should use API keys to monitor usage by third-party developers and leverage a stronger mechanism for authorization, such as signed API calls or OAuth.

Q: How can I address or prevent API threats or abuse?

Amazon API Gateway supports throttling settings for each method in your APIs. You can set a standard rate limit and a burst rate limit per second for each method in your REST APIs. Further, Amazon API Gateway automatically protects your backend systems from distributed denial-of-service (DDoS) attacks, whether attacked with counterfeit requests (Layer 7) or SYN floods (Layer 3).

Q: Can Amazon API Gateway work within an Amazon VPC?

No. Amazon API Gateway endpoints are always public to the Internet. Proxy requests to backend operations also need to be publicly accessible on the Internet. However, you can generate a client-side SSL certificate in Amazon API Gateway to verify that requests to your backend systems were sent by API Gateway using the public key of the certificate.

Q: Can I verify that it is API Gateway calling my backend?

Yes. Amazon API Gateway can generate a client-side SSL certificate and make the public key of that certificate available to you. Calls to your backend can be made with the generated certificate, and you can verify calls originating from Amazon API Gateway using the public key of the certificate.

Q: Can I use AWS CloudTrail with Amazon API Gateway?

Yes. Amazon API Gateway is integrated with [AWS CloudTrail](https://aws.amazon.com/cloudtrail/) to give you a full auditable history of the changes to your REST APIs. All API calls made to the Amazon API Gateway APIs to create, modify, delete, or deploy REST APIs are logged to CloudTrail in your AWS account.

[back to top >>](https://aws.amazon.com/api-gateway/faqs/)

## **Managment, Metrics, and Logging**

Q: How can I monitor my Amazon API Gateway APIs?

Amazon API Gateway logs API calls, latency, and error rates to Amazon CloudWatch in your AWS account. The metrics are also available through the Amazon API Gateway console in a REST API dashboard. API Gateway also meters utilization by third-party developers, the data is available in the API Gateway console and through the APIs.

Q: Can I set up alarms on the Amazon API Gateway metrics?

Yes, Amazon API Gateway sends logging information and metrics to Amazon CloudWatch. You can utilize the Amazon CloudWatch console to set up custom alarms.

Q: How can I set up metrics for Amazon API Gateway?

By default, Amazon API Gateway monitors traffic at a REST API level. Optionally, you can enable detailed metrics for each method in your REST API from the deployment configuration APIs or console screen. Detailed metrics are also logged to Amazon CloudWatch and will be charged at the CloudWatch rates.

Q: Can I determine which version of the API my customers are using?

Yes. Metric details are specified by REST API and stage. Additionally, you can enable metrics for each method in your REST API.

Q: Does Amazon API Gateway provide logging support?

Yes. Amazon API Gateway integrates with Amazon CloudWatch Logs. You can optionally enable logging for each stage in your API. For each method in your REST APIs, you can set the verbosity of the logging, and if full request and response data should be logged.

Q: How quickly are logs available?

Logs, alarms, error rates and other metrics are stored in Amazon CloudWatch and are available near real time.

[back to top >>](https://aws.amazon.com/api-gateway/faqs/)

## **Throttling and Caching**

Q: How can I protect my backend systems and applications from traffic spikes?

Amazon API Gateway provides throttling at multiple levels including global and by service call. Throttling limits can be set for standard rates and bursts. For example, API owners can set a rate limit of 1,000 requests per second for a specific method in their REST APIs, and also configure Amazon API Gateway to handle a burst of 2,000 requests per second for a few seconds. Amazon API Gateway tracks the number of requests per second. Any requests over the limit will receive a 429 HTTP response. The client SDKs generated by Amazon API Gateway retry calls automatically when met with this response.

Q: Can I throttle individual developers calling my APIs?

Yes. With usage plans you can set throttling limits for individual API keys.

Q: How does throttling help me?

Throttling ensures that API traffic is controlled to help your backend services maintain performance and availability.

Q: At which levels can Amazon API Gateway throttle inbound API traffic?

Throttling rate limits can be set at the method level. You can edit the throttling limits in your method settings through the Amazon API Gateway APIs or in the Amazon API Gateway console.

Q: How are throttling rules applied?

First. API Gateway checks against your AWS account limit. If the traffic is below the set account limit, API Gateway checks the limit you have set on a stage or method. If the traffic is below the stage limit, then API Gateway applies the usage plans limits you set on a per-API key basis.

Q: Does Amazon API Gateway provide API result caching?

Yes. You can add caching to API calls by provisioning an Amazon API Gateway cache and specifying its size in gigabytes. The cache is provisioned for a specific stage of your APIs. This improves performance and reduces the traffic sent to your back end. Cache settings allow you to control the way the cache key is built and the time-to-live (TTL) of the data stored for each method. Amazon API Gateway also exposes management APIs that help you invalidate the cache for each stage.

Q: What happens if a large number of end users try to invoke my API simultaneously?

If caching is not enabled and throttling limits have not been applied, then all requests will pass through to your backend service until the account level throttling limits are reached. If throttling limits are in place, then Amazon API Gateway will shed the necessary amount of requests and send only the defined limit to your back-end service. If a cache is configured, then Amazon API Gateway will return a cached response for duplicate requests for a customizable time,&nbsp;but only if under configured throttling limits. This balance between the backend and client ensures optimal performance of the APIs for the applications that it supports. Requests that are throttled will be automatically retried by the client-side SDKs generated by Amazon API Gateway. By default, Amazon API Gateway does not set any cache on your API methods.

Q: How do APIs scale?

Amazon API Gateway acts as a proxy to the backend operations that you have configured. Amazon API Gateway will automatically scale to handle the amount of traffic your API receives. Amazon API Gateway does not arbitrarily limit or throttle invocations to your backend operations and all requests that are not intercepted by throttling and caching settings in the Amazon API Gateway console are sent to your backend operations.

[back to top >>](https://aws.amazon.com/api-gateway/faqs/)

## **Billing**

Q: How am I charged for using Amazon API Gateway?

Amazon API Gateway rates are $3.50 per million API calls, plus the cost of data transfer out, in gigabytes. If you choose to provision a cache for your API, hourly rates apply. Please see the [API Gateway Pricing](https://aws.amazon.com/AWSLambdaResourcesEdit/Api-gateway/api-gateway/pricing/) pages for details on data transfer and caching costs.

Q: Who pays for Amazon API Gateway API calls generated by third-party developers?

The API owner is charged for the calls to their APIs on API Gateway.

Q: If an API response is served by cached data, is it still considered an API call for billing purposes?

Yes. API calls are counted equally for billing purposes whether the response is handled by your backend operations or the Amazon API Gateway caching operation.